Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores (II/2018)

Tarea 2

Fecha de entrega: lunes 27 de agosto de 2018 a las 11:59 AM

Parte teórica

Pregunta 1

El **bit de paridad** es un mecanismo que se utiliza con el propósito de detectar errores durante la transmisión de información binaria. Consiste en un bit extra incluido en un mensaje binario para hacer que el número de 1s en él sea par. Una vez transmitido el mensaje, incluido el bit de paridad, se verifica en el receptor si la cantidad de bits en el mensaje es par o impar. En caso de que sea impar, el mensaje fue transmitido con un error, por lo tanto debe ser descartado.

a) Usando combinaciones de las compuertas $\wedge, \vee, \oplus,$ y $\neg$, construya los circuitos para un generador de bit de paridad para mensajes de 3 bits, y para su verificador de mensajes asociado. Este último debe indicar mediante un 1 si el mensaje es correcto y con un 0 si viene con errores. ¿Cuántos bits con errores es capaz de detectar este esquema?

b) Usando sólo combinaciones de uno de los siguientes tipos de compuerta, $\wedge, \vee, \oplus$ y $\neg$, construya los mismos circuitos del ítem anterior. Es decir, si por ejemplo escoge hacerlo con $\wedge$, sólo puede emplear ese tipo de compuertas en su diseño. No es necesario que ambos circuitos utilicen el mismo tipo de compuerta.

c) Generalice el diseño del ítem anterior, para mensajes de N bits, con $N >= 3$.

Pregunta 2

Un comparador de números es un circuito que, dados dos números A y B en representación posicional, indica cuál es el mayor, o si estos son iguales. El circuito posee tres salidas, donde la primera entrega un 1 sólo si A es el mayor, la segunda un 1 sólo si ambos son iguales, y la tercera un 1 sólo B es mayor.

a) Diseñe un comparador de números naturales de N bits.

b) Diseñe un comparador de números enteros de N bits.

c) Diseñe un comparador del valor absoluto de números enteros de N bits.

Parte programación

Escriba un programa que, dada la descripción de un circuito lógico (binario), que utiliza compuertas $\wedge, \vee, \oplus$ y $\neg$, debe producir la descripción de un circuito equivalente, es decir, con las mismas **entradas**, **salidas** y comportamiento, pero que utiliza exclusivamente compuertas $\overline{\wedge}$ (`NAND`, la negación de `AND`).

El programa debe ser escrito en Python 3.5, debe tener como nombre `logic_translator.py` y debe tomar dos argumentos: el primero correspondiente a la ruta a un archivo `JSON` que describe el circuito a traducir, y el segundo, la ruta a un nuevo archivo `JSON` que será escrito por su programa y corresponderá al nuevo circuito equivalente.

A continuación, se muestra un ejemplo del archivo de entrada `JSON` que podría recibir su programa:

```
{
    "circuit": [
    {"from": "input1", "to": "xor1"},
    {"from": "input1", "to": "and1"},
    {"from": "input2", "to": "xor1"},
    {"from": "input2", "to": "and1"},
    {"from": "xor1", "to": "out1"},
    {"from": "and1", "to": "out2"}
    ]
}
```

Este corresponde al siguiente circuito:

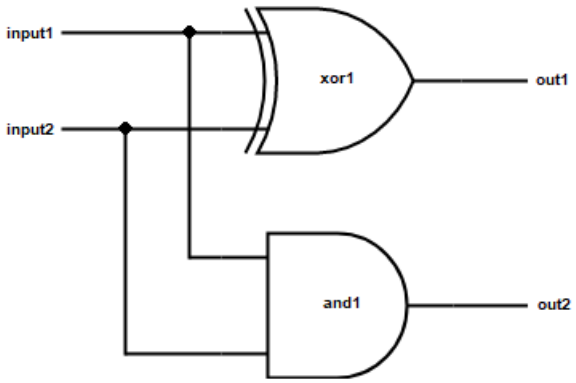

Figura 1: Circuito detallado en el ejemplo de archivo `JSON`.

La convención de entradas y salidas será "`input`$\{i\}$" y "`out`$\{j\}$" respectivamente, siendo i y j sus numeraciones para diferenciar las señales entre sí, tal como se aprecia en el código que representa a la figura 1. Por otra parte, la convención de las compuertas es simplemente el uso de sus nombres con su respectiva numeración: "`and`$\{k\}$", "`or`$\{l\}$", "`xor`$\{m\}$", "`not`$\{n\}$". Finalmente, el archivo `JSON` generado **debe** incluir compuertas del tipo $\overline{\wedge}$ haciendo uso de la convención "`nand`$\{s\}$", siendo s la numeración correspondiente.

Parte práctica

Utilizando el proyecto base de Vivado llamado **Proyecto Base**, que puede encontrar en el Syllabus, construya un componente VHDL que sea capaz de calcular el producto punto entre dos vectores.

Descripción del componente

El componente a armar debe recibir como entrada dos buses de datos de 16 bits, que representan a los vectores, y como salida debe entregar un bus de datos de 16 bits con el resultado final, que corresponde a un escalar. El componente debe estar instanciado en el archivo `Basys3` y estar conectado a las entradas y salidas mencionadas más adelante.

Representación de los vectores

Para la construcción, deberán utilizar una representación de 16 bits, tal que cada valor usa 4 bits para su representación. O sea, si tenemos el siguiente vector binario:

$$[\ 0101101010110111\ ]$$

Lo podemos dividir en cuatro números para tener un vector de tamaño 4:

$$\begin{bmatrix} 0101 \\ 1010 \\ 1011 \\ 0111 \end{bmatrix}$$

Que finalmente en base decimal corresponde al vector:

$$\begin{bmatrix} 5 \\ 10 \\ 11 \\ 7 \end{bmatrix}$$

Ingreso de los vectores

Para ingresar los valores de los vectores se usarán los *switches* y botones. Para ingresar un vector, se debe ingresar la representación binaria deseada en los *switches* y al apretar el botón derecho, se ingresa uno de los vectores. Para el segundo es el mismo proceso, pero con el botón izquierdo. Para guardarlos, pueden hacer uso del componente del registro disponible en el **Proyecto Base**.

Presentación del resultado

El escalar resultante se debe mostrar en los *displays* de la placa.

Funcionalidades de la placa

A continuación, se muestra un diagrama con todas las funcionalidades de los elementos de la placa para esta tarea:

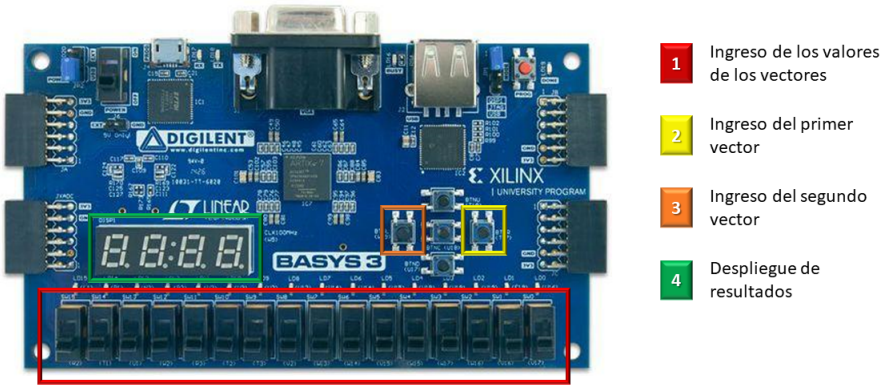

Figura 2: Diagrama de uso de la placa para el ingreso de los valores de los vectores.

Entrega y evaluación

La tarea se debe realizar de **manera individual** y la entrega se realizará a través de GitHub. El formato de entrega para cada modalidad será:

- **Teórica:** Archivo llamado `tarea2.tex`, junto con los recursos necesarios para que este pueda ser compilado correctamente.
- **Programación:** Archivo `logic_translator.py` y los extras necesarios para ejecutar su programa correctamente.
- **Práctica:** Archivo `.bit` sintetizado para programar la placa, además del proyecto de Vivado que pueda generar la síntesis.

El repositorio debe contar con un `README.md` escrito en *Markdown* que identifique sus datos, la versión de la tarea realizada y consideraciones que el corrector deba tomar en cuenta, tales como el número de placa utilizado, sistema operativo usado para realizar la tarea, paquetes de LaTeX empleados, etc.
Archivos que no compilen o no cumplir este formato de entrega implicará nota **1.0** en la tarea. En caso de atraso, se aplicará un descuento de **1.0** punto por cada 6 horas o fracción.

Política de Integridad Académica

Los alumnos de la Escuela de Ingeniería deben mantener un comportamiento acorde al Código de Honor de la Universidad:

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

En particular, se espera que mantengan altos estándares de honestidad académica. Cualquier acto deshonesto o fraude académico está prohibido; los alumnos que incurran en este tipo de acciones se exponen a un procedimiento sumario. Específicamente, para los cursos del Departamento de Ciencia de la Computación, rige obligatoriamente la siguiente política de integridad académica. Todo trabajo presentado por un alumno (grupo) para los efectos de la evaluación de un curso debe ser hecho individualmente por el alumno (grupo), sin apoyo en material de terceros. Por "trabajo" se entiende en general las interrogaciones escritas, las tareas de programación u otras, los trabajos de laboratorio, los proyectos, el examen, entre otros. Si un alumno (grupo) copia un trabajo, los antecedentes serán enviados a la Dirección de Docencia de la Escuela de Ingeniería para evaluar posteriores sanciones en conjunto con la Universidad, las que pueden incluir reprobación del curso y un procedimiento sumario. Por "copia" se entiende incluir en el trabajo presentado como propio partes hechas por otra persona. Está permitido usar material disponible públicamente, por ejemplo, libros o contenidos tomados de Internet, siempre y cuando se incluya la cita correspondiente.